In [1]:
```
import numpy as np
```

In [2]:
```
states = ["Alabama","Alaska","Arizona","Arkansas","California","Colorad
o","Connecticut","Delaware","District of Columbia","Florida",
"Georgia","Hawaii","Idaho","Illinois","Indiana","Iowa","Kansas","Kentuck
y","Louisiana","Maine","Maryland","Massachusetts",
"Michigan","Minnesota","Mississippi","Missouri","Montana","Nebraska","Ne
vada","New Hampshire","New Jersey","New Mexico",
"New York","North Carolina","North Dakota","Ohio","Oklahoma","Oregon","P
ennsylvania","Rhode Island","South Carolina",
"South Dakota","Tennessee","Texas","Utah","Vermont","Virginia","Washingt
on","West Virginia","Wisconsin","Wyoming"]
votes = [9,3,11,6,55,9,7,3,3,29,16,4,4,20,11,6,6,8,8,4,10,11,16,10,6,10,
3,5,6,4,14,5,29,15,3,18,7,7,20,4,9,3,11,38,6,3,13,12,5,10,3]
```

In [3]:
```
#a. use dynamic programming to finds the total number of ways
ways = [[0 for i in range(270)] for j in range(51)]
for i in range(51):
    ways[i][0] = 1
ways[0][votes[0]] = 1

for i in range(1, 51):
    for j in range(1, 270):
        p1 = ways[i - 1][j]
        p2 = 0
        if ((j - votes[i]) >= 0):
            p2 = ways[i - 1][j - votes[i]]
        ways[i][j] = p1 + p2

print("The total number of ways to achieve tie is " + str(ways[50][269])
+ ".")
```

```
The total number of ways to achieve tie is 16976480564070.
```

In [4]:
```
#Justification for a.
#Following the linear programming method, the number of ways to sum to S
within n first elements is
#the number of ways to sum to S within n-1 first elements(excluding the
 nth element) plus the number of ways
#to sum to S-(nth element) within n-1 first elements(including the nth e
lement).

#The algorithm starts with
```

In [21]:
```
#b. find a combination
subset = [[set() for i in range(270)] for j in range(51)]
subset[0][votes[0]] = {0}

for i in range(1, 51):
    for j in range(1, 270):
        p1 = subset[i - 1][j].copy()
        p2 = set()
        if ((j - votes[i]) >= 0):
            p2 = subset[i - 1][j - votes[i]].copy()
        if p1:
            subset[i][j] = p1
        elif p2:
            p2.add(i)
            subset[i][j] = p2
if subset[50][269]:
    states_chosen = []
    s = 0
    for j in subset[50][269]:
        states_chosen.append(states[j])
        s += votes[j]
    print("A way to achieve tie is " + str(states_chosen) + ".")
else:
    print("Not feasible.")
```

```
A way to achieve tie is ['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'C
alifornia', 'Colorado', 'Connecticut', 'Delaware', 'District of Columbi
a', 'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana', 'Io
wa', 'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland', 'Massachus
etts', 'Michigan', 'Minnesota'].
269
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23}
```

In [108]:
```
#Justification for b.
#As a modified version of a., the algorithm stores the combination direc
tly in a set. For either comtaining
#current element or not, if there exists a solution, or solutions, subse
t[i, s] is either subset[i-1, s] or
#subset[i-1, s-(nth element)] union nth element.
```

In [ ]:
```
#d. runtime analysis and justification
#For dynamic programming in part a and b, the dimension of the 2D array
 is
#number of states(n) * number of total votes(K).
#And the algorithm visits every grids once and compare one or two values
in other grids, counting up to O(2nK)=O(nK)
```